LA-UR-19-24736
Approved for public release; distribution is unlimited.

Title: LANSCE Diagnostic Robot Localization

Author(s): Montoya, Lucas Sigfredo

Intended for: Master thesis manuscript

Issued: 2019-05-22

Lucas Montoya
Candidate

Mechanical Engineering
Department

This thesis is approved, and it is acceptable in quality and form for publication:

Approved by the Thesis Committee:

Christopher Hall , Chairperson

Svetlana Poroseva

Meeko Oishi

LANSCE DIAGNOSTIC ROBOT LOCALIZATION

by

Lucas Montoya

B.S., Mechanical Engineering, New Mexico State University, 2014

THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science
Mechanical Engineering

The University of New Mexico
Albuquerque, New Mexico

May 2019

Acknowledgements

I would like to thank my mentor James Sedillo for his untiring support throughout this whole project, facilitating a good working environment and providing me with valuable insight throughout my research. David Bonal of National Instruments deserves a thanks as he graciously provided me with the LabVIEW robotics module, enabling the capability for simulation. Ray Roybal for providing me with CAD files of the beam tunnel for simulating the robot in. To Los Alamos National Laboratory for allowing me to work on this project during my internship, and everyone there who provided assistance whenever I needed it. Finally, I would like to thank Professor Christopher Hall for his advisement during the entire process.

LANSCE Diagnostic Robot Localization

Lucas Montoya

B.S., Mechanical Engineering, New Mexico State University, 2014

M.S., Mechanical Engineering, University of New Mexico, 2019

Abstract

Los Alamos Neutron Science Center (LANSCE) operates a linear particle accelerator (LINAC) that is used for a number of scientific research projects. Due to the presence of harmful radiation, researchers are not allowed in the beam tunnel during operation. Since the beam tunnel is inaccessible, an autonomous mobile robot is to be developed and deployed in the tunnel, monitoring the accelerator during operation. The robot will present real time data for operators and scientists from sensors such as a video camera, thermal camera, microphones, and muon detectors; allowing for beam diagnostics previously unavailable.

Localization is a fundamental step in autonomous navigation of mobile robots, answering the “where am I” question. For this project, an Extended Kalman Filter (EKF) algorithm is implemented as a positional estimator. The EKF relies on an odometric motion model to predict the robot’s position and LIDAR measurement data to update the robot’s position. The measurement model is based on

features extracted from the LIDAR point cluster using the Split-and-Merge algorithm, and the nearest neighbor algorithm associates the features to an *a priori* feature map of the beam tunnel. The predicted and measured positions are then joined using a weighted value by means of the Kalman gain. The effectiveness of this localization technique is demonstrated using the LabVIEW robotics simulator.

Contents

List of Figures

List of Abbreviations

CAD	Computer Aided Design
EKF	Extended Kalman Filter
ICNN	Individual Compatibility Nearest Neighbor
LANSCE	Los Alamos Neutron Science Center
LIDAR	Light Detection and Ranging
LINAC	Linear Accelerator
SEF	Successive Edge Following
SLAM	Simultaneous Localization and Mapping
TOF	Time-of-Flight

Chapter 1

Introduction

Mobile robotics have long been used in hazardous situations with an operator controlling the robot remotely, acting as eyes and ears to an uninhabitable environment. A trend has been moving towards fully autonomous robots being used in hazardous environments, and to be able to realize full autonomy, it is necessary to self-localize within that environment.

Mobile robotics self-localization asks the question of “where am I”, which is the answer our research aims to solve in order for the robot to successfully navigate and complete its designated mission. An overview of the sensors used, robot model, data acquisition, localization strategy, and the simulation model will be presented.

1.1 Background

The Los Alamos Neutron Science Center (LANSCE) user facility is home to an 800 MeV linear accelerator (LINAC), supplying multiple experimental areas with intense pulsed proton beam travelling 84 percent the speed of light. With the accelerator, LANSCE supports many different programmatic and scientific needs for defense and civilian research. With over a thousand user visits to the facility annually [1], and the production of medical isotopes used for cardiac imaging for

over 30,000 patients a month [2], it is crucial to ensure reliable operation of the accelerator.

There are numerous systems working to keep the machine running during production that frequently fail and cause unnecessary downtime. Currently there are limited diagnostic tools used to monitor the status of these systems. It is of interest to gather as much information as possible to better predict and detect failures and help facilitate tuning of the beam during startup. Predictors of failure include elevated radiation levels, which indicate beam spill, and excess heat given off by components along the beam line, both of which can be measured using the proposed robot.

Because of the radiation produced by proton interactions, it renders the beam tunnel inaccessible by personnel during production, hindering direct measurements that can be taken to provide beam status. With the use of a mobile robot in the tunnel, measurements can be taken safely from a remote location. To avoid the requirement of an operator taking data on a regular schedule, it makes sense to design the robot to be fully autonomous, reporting its findings after each tour of the beam tunnel.

To achieve full autonomy a mobile robot must first be capable of self-localization. It is important so that it can navigate the tunnel accurately, providing reliable data from each measurement.

1.1.1 Accelerator Environment

The linear accelerator tunnel is ¾ miles long and underground which houses the accelerator structures and supporting equipment. A cross section diagram of the tunnel can be seen in figure 1.1, depicting where the robot is deployed relative to the accelerator inside the tunnel.

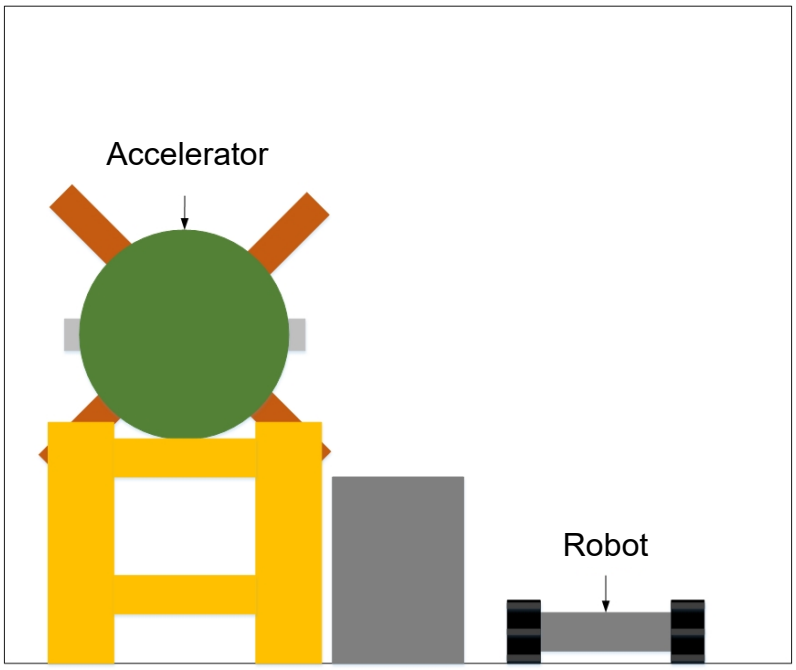


Figure 1.1: *Accelerator Tunnel X-Section*

Because of the radiation and location of the accelerator underground, there are many design considerations to account for to help mitigate or eliminate potential issues. These issues will be discussed in later sections as they pertain to different aspects of the robot and in the localization strategy.

1.2 Research Overview

The purpose of this research is to develop a localization strategy for a differential drive robot in a known indoor environment with sparse landmark features. To accomplish this, the Extended Kalman Filter (EKF) is employed. The EKF is a recursive algorithm that relates the current state to the previous estimate of the state using the state transition and measurement models. Because neither the state transition or measurement models are linear, Jacobians are used to linearize them such that Gaussian approximation is maintained. [3]

To implement the EKF properly, the kinematics of the robot must be evaluated to develop the state transition model. Landmark acquisition is developed for the measurement update, and data acquisition algorithms are developed to minimize spurious measurements.

The majority of this work is developed and tested using the LabVIEW Robotics Simulator environment as the beam tunnel has access requirements severely limiting real-world testing opportunities. The tunnel and robot chassis are modeled and imported into the robotics simulator to produce accurate results resembling real world operation in the tunnel.

1.3 Outline of Thesis

In chapter 2 the literature studied is reviewed. Articles establishing sensor selection, along with the feature extraction and data association techniques given

different applications is covered. An overview of localization strategies is presented, with special attention given to the EKF localization technique.

The vehicle model of the robot is covered in chapter 3. The chassis and sensors selected are described. Kinematics equations of a rigid differential drive robot are formulated and expressed in vector form. Odometry is established using the kinematic model and encoders.

Landmark acquisition is addressed in chapter 4, which makes use of multiple algorithms to deduce sensor data into something useful. Line extraction is performed utilizing successive edge following and Split-and-Merge algorithms. Landmarks such as corners, walls, and beam structures are then determined from the extracted line geometries.

In chapter 5, the Extended Kalman Filter is discussed, and is presented in three segments. The prediction stage is first, and is based on the motion model of the robot. The next is the measurement update step, which is based on the measurement model, providing corrections to the state estimation and error covariance based on sensor observations. Data association ties the two together, in the form of the nearest neighbor algorithm, utilizing shared matrices comparing the measured landmarks to the map of the beam tunnel.

In chapter 6 the LabVIEW Robotics simulator is reviewed. The beam tunnel and robot are modeled and imported into the simulator to best mimic conditions seen in the tunnel to provide an accurate representation of real obstacles. Results are

then gathered and analyzed after collecting robot tracking data as it traverses the tunnel in the simulation.

A summary of the research is presented in chapter 7, along with future plans for the project beyond this thesis.

Chapter 2

Literature Review

In this chapter an overview of the process and reasoning behind the decision to use the Extended Kalman Filter (EKF) as the state estimator is examined. In conjunction with the EKF, sensors on the robot are assessed for their viability in the system. Supporting algorithms such as line extraction and data association are also reviewed for their integration into the localization strategy presented.

2.1 Mobile Robot Localization

There are numerous localization strategies that can be implemented, as was discussed by Jensfelt [4]. Of these approaches, the Extended Kalman Filter (EKF) is the most advantageous due to the fact that the EKF is reliant on a known map, or hypothesis of landmarks in the environment. Since a map of the beam tunnel can be defined from CAD drawings, a mapping algorithm such as the Simultaneous Localization and Mapping (SLAM) techniques [5, 6] are not necessary.

The EKF is a landmark based localization algorithm that fuses information from multiple sensors, such as the LIDAR and encoder odometry data. The LIDAR measurement model and encoder odometry model are linearized as they are non-linear, to provide Gaussian estimates through the Kalman filter. [7]

The EKF consists of two major steps in the estimation of the robot state. The first step is predicting the robot state obtained from odometric data. Odometry is the measurement of distance travelled using data obtained from sensors, specifically encoders in our application, which are coupled to the motors. Motor rotation measurements can then be used to calculate the motion of the robot using the kinematic model. The prediction is also used to estimate where the landmarks should be based on the aforementioned odometry. When a landmark is observed, the difference is taken between the observed and predicted position, referred to as the innovation [5]. Because odometry is notorious for producing error, a higher confidence is placed upon the observation model from the LIDAR sensor. The Kalman gain is employed to weigh the observation data as more reliable [8, 9]. The EKF is recursive such that multiple observations can be processed and only the last estimate is necessary to provide the current state estimate.

2.2 Sensors

There are several types of sensors used on mobile robotics for the purpose of localization. Types of sensors used in mobile robotics range from GPS, magnetic compasses, dead reckoning, guide-path-following, and time-of-flight (TOF) range finding sensors [10]. The selection process of the various types of sensors are reviewed, examining the feasibility and integration into the robot system.

Given the robot's environment, some sensors are not a viable option. The tunnel is underground and encased in concrete, thus GPS is not an option as the signal cannot penetrate into the tunnel. High powered magnets are used to focus the proton beam, which would distort magnetic compass readings rendering its data unreliable. During accelerator maintenance periods, equipment is rearranged which can obstruct guide paths and limit the flexibility of the robot to be able to take data effectively given different missions through its use. TOF and dead reckoning sensors are left as practical options for the robot.

Sensors that measure the time taken for emitted energy to travel to a target and reflect back is considered to be of the TOF family of ranging sensors [10], such as light detection and ranging (LIDAR) or ultrasonic sensors. Given the resolution and accuracy that LIDAR can provide over ultrasonic sensors, the LIDAR was selected as the primary measurement device.

In addition to the LIDAR, the motor's magnetic encoders are incorporated to aid in localizing the robot; providing necessary data for dead reckoning measurements. The magnetic encoders use Hall Effect sensors that pick up the presence of a ferrous metal gear tooth. With each passing tooth, a count can be made that relates to an angular displacement of the motors. Advantages of using a magnetic encoder include durability to vibrations caused by robot motion [11]. The disadvantage is that there is no way to determine direction since it is not a quadrature encoder. The sensors best fitting for this project are TOF and dead reckoning, due to environmental constraints, range, and the accuracy necessary for localization.

2.3 Feature Extraction Methods

The selected feature extraction method is reliant on the LIDAR sensor point cloud output. From the LIDAR point cloud, geometries can be extracted that represent the environment accurately. Nguyen *et al.* details many of the various methods to extract features from the environment [12]. Through the comparisons made from Nguyen's research, an effective feature extraction strategy was developed. Nguyen describes employing a clustering algorithm to filter out noisy points and segments the points into adjacent clusters before implementing the various line extraction techniques known as Successive Edge Following (SEF).

To further understand the SEF, Siadat *et al.* [13] described the algorithm, and performed a comparison between the SEF and other segmentation methods in different environment representations such as complex and structured indoor/outdoor environments. Since the beam tunnel best represents a structured indoor environment, those results are the most applicable. In Siadat's research the SEF excelled in both computational time and in reducing the amount of error. The SEF algorithm segments points into contiguous clusters by taking the distance between adjacent points and comparing it to a threshold distance. If the Euclidean distance between adjacent points is greater than the threshold, the point cluster is segmented. However, in using LIDAR, the distance between adjacent points will be greater the further the sensed object is due to the magnification of distance from angular measurements. Yan Bu *et al.* [14] solve this problem by incorporating an adaptive threshold which is reliant on the mean polar radius of the point cloud, and the angle between adjacent points.

In Nguyen's research [12] the most successful algorithm for defining line segments was the Split-and-Merge algorithm, also known as the Ramer-Douglas-Puecker or iterative end point algorithm [15]. The data was taken in a structured indoor environment. The Split-and-Merge algorithm performed the quickest and provided the most reliable and precise data as can be seen in Figure 2.2.

Algorithm	**Complexity**	**Speed** $[Hz]$	**N.Lines**	**Correctness**		**Precision**	
				TruePos $[\%]$	FalsePos $[\%]$	$\sigma_{\Delta r}$ $[cm]$	$\sigma_{\Delta \alpha}$ $[deg]$
Split-Merge + Clus.	$N \times logN$	1470	641	86.0	8.9	1.95	0.74
Incremental	$S \times N^2$	344	561	77.8	5.9	2.04	0.72
Incremental + Clus.		617	567	79.2	5.1	2.04	0.76
Line Regression	$N \times N_f$	364	577	76.4	10.1	1.99	0.80
LR + Clus.		384	562	75.8	8.4	1.97	0.79
RANSAC	$S \times N \times N.Trials$	29	749	75.6	31.5	1.68	0.77
RANSAC + Clus.		93	547	70.7	12.2	1.37	0.70
Hough Transform	$S \times N \times NC + S \times NR \times NC$	8	825	82.0	32.5	1.63	0.76
HT + Clus.		9	600	79.5	10.0	1.51	0.67
EM	$S \times N1 \times N2 \times N$	0.6	1153	78.6	53.7	2.09	0.97
EM + Clus.		0.7	709	80.3	23.1	1.58	0.73

Figure 2.1: *Line Extraction Algorithm Performance by Nguyen et al [12]*

Split-and-merge is a well-known and widely used recursive algorithm that takes the first and last points from a cluster and calculates the furthest perpendicular distance point in the cluster from the line segment. If the perpendicular distance is greater than a threshold distance, the algorithm will split the cluster in two and continue recursively until no more splits are made [12, 16, 17]. From the Split-and-Merge we are left with only the line segments as perceived by the LIDAR.

2.4 Data Association

Between the map and observation model of the robot, we must determine which landmark the observed feature corresponds to. Cooper [18] performed a comparison of various data association techniques in EKF-SLAM applications, most notably the Individual Compatibility Nearest Neighbor (ICNN) algorithm, which shares components with the EKF.

The ICNN is based on establishing the compatibility between an observation and landmarks using Mahalonobis distance. Mahalonobis distance is the distance of an observation from a distribution set [19].

$$D_M{}^2 = (\vec{x} - \vec{y})^T S^{-1} (\vec{x} - \vec{y}) \tag{2.1}$$

Where $\vec{x}$ and $\vec{y}$ are random vectors, and S is the associated covariance. In this definition the Mahalonobis distance is also known as the generalized squared interpoint distance, and describes a similarity measure [20]. In our case we have observations and the landmark map as the vectors. The ICNN deems the pairing with the smallest Mahalonobis distance as compatible if it is below a Chi-Squared distribution confidence level threshold criterion as the Mahalonobis follows a Chi-Squared distribution. Because ICNN is a greedy algorithm [21], never reevaluating a pairing with a more compatible landmark, an additional function will be developed to ensure we are associating the correct landmark.

2.5 Summary

A review of localization strategies and associating topics has been presented, from which a strategy can be derived to suit our needs. It has been demonstrated that while mobile robot localization has been a topic of focus, there are still faults of various approaches, such as EKF divergence, or misassociations of data association techniques that have been investigated before implementation.

Chapter 3

Vehicle Model

In this chapter we discuss the vehicle model, which includes the chassis, sensors, inverse kinematics, and odometry. The first two sections provide the hardware specifications, with the last two describing the motion using inverse and forward kinematics.

3.1 Chassis

The chassis used in the simulation is a Dr. Robot Jaguar Lite chassis, a tracked mobile robot platform, equipped with 24V DC motor actuating each track and integrated magnetic encoders [22]. The chassis is modeled in SolidWorks and imported into the LabVIEW robotics simulator as shown in figure 3.1.

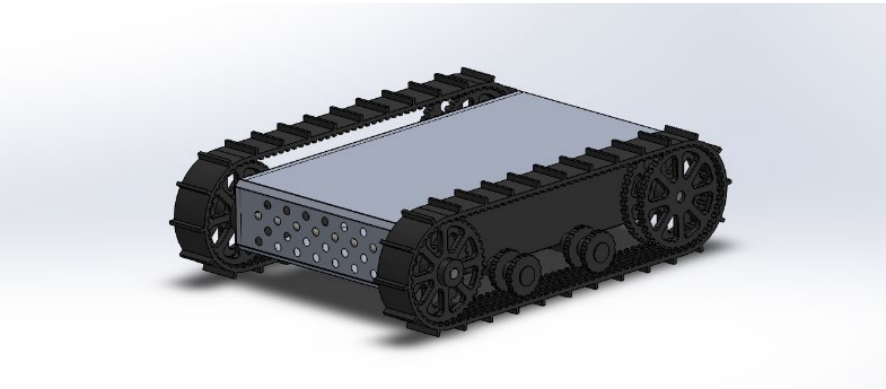


Figure 3.1: *Robot Chassis CAD Model*

The Dr. Robot platform was inherited from a previous project [23] and is made for rugged outdoor terrain. Due to the rugged terrain design of the robot, the paddled tracks induce vibrations into the chassis causing some slipping, adding error to the odometry model.

3.1.1 Sensors

In order to achieve localization, the robot needs to be provided with sensors suitable for its environment. It is crucial the sensors are selected appropriately so we can detect the robot's surroundings accurately and repeatedly with as little error as possible. In this section we discuss the sensors employed to accomplish localization and why the sensors used were selected.

3.1.1.1 LIDAR

In order for the robot to localize itself, it needs a way to sense the environment around it. Hokuyo URG-04LX-UG01 LIDAR sensor was selected to be used due to its small size and device driver support provided in the LabVIEW robotics simulator.

Figure 3.2: *Hokuyo URG-04LX-UG30 LIDAR [24]*

The Hokuyo LIDAR sensor has a scan angle of 240° and angular resolution of 0.36°, which gives us 667 data points per scan. The sensing range of the sensor is 4 meters with an accuracy of 3% of the detected distance and a resolution of 1mm. The sensor performs scans at 10 Hz, which is fast enough, relative to the robot velocity, that it doesn't miss potential obstacles [24].

LIDAR determines range by measuring the phase shift between the emitted laser signal, and the reflected signal. Because the laser possesses the properties of light, it also obeys the formula for light as shown in Equation 3.1, where c is the speed of light (meters/second), lambda is the wavelength (meters), and f is the frequency (Hertz).

$$c = \lambda f \tag{3.1}$$

The laser is transmitted at a known frequency, and the speed of light is constant, thus the wavelength can be found. The wavelength is necessary to determine the

distance D in meters, travelled by the laser signal as shown in Equation 3.2, where θ is the phase difference between the transmitted laser and reflected laser signals. [25]

$$D = \frac{\lambda}{4\pi}\theta \tag{3.2}$$

For the LIDAR to perform a scan, the laser is pulsed onto a rotating mirror. Each reflected pulse represents a point in the scan. LIDAR range-finding schematic is shown in figure 3.3.

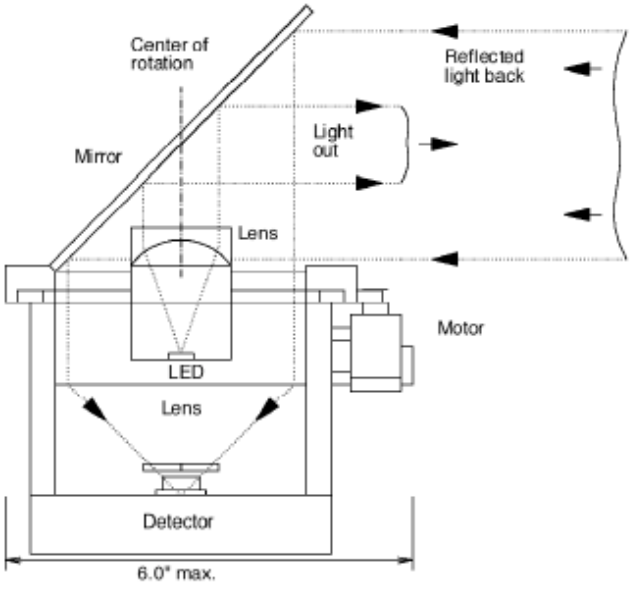


Figure 3.3: *LIDAR Range-finding Schematic [25]*

The LIDAR sensor was simulated in the LabVIEW robotics module as if the hardware were present, utilizing the sensor's communication protocol SCIP2.0 to acquire data scans. Software was available through the robotics module, although some adjustments to the code were necessary to obtain the 10 Hz scanning frequency.

3.1.1.2 Encoders

Using the LabVIEW robotics simulator, absolute encoders are provided, that outputs the shaft position in radians, from which the motion is deduced by taking the change of position as opposed to a count. With each angular difference, an angular velocity is obtained, which can be related to a robot body velocity given the gear ratio and cog radius. The kinematic model is to be covered in the following section 3.2.

In practice incremental rotary magnetic encoders are used, one mated to each DC motor to provide information regarding the robot motion. Given that these encoders are not quadrature, direction is not inherently known from the device, and will be assumed based on the control input; however within the simulation it was not an issue.

3.2 Inverse Kinematics

The main goal of the inverse kinematics is to transform the desired robot frame velocities to motor angular velocities. The equations that describe the system are derived from Kelly's constrained kinematics for mobile robots [26]. LabVIEW robotics module software provides built in functions to accomplish the kinematic model, but a quick overview is presented.

3.2.1 Constraints

Differential drive motion is characterized by two non-holonomic constraints and is derived by making two assumptions. The first constraint is a pure rolling condition, which means that each track can roll forward or backward in the direction of the track velocity without slipping against the floor contact surface. The track velocity can be calculated as shown in Equation 3.3, where r is the cog radius, $\dot{\varphi}$ is the angular velocity, and v is the track velocity [27].

$$\begin{bmatrix} v_R \\ v_L \end{bmatrix} = \begin{bmatrix} \frac{r\dot{\varphi}_R}{g} \\ \frac{r\dot{\varphi}_L}{g} \end{bmatrix} \tag{3.3}$$

The second non-holonomic constraint is that the tracks cannot move laterally, meaning that $\dot{x}_r$ is always zero in the local robot frame. The matrix form is shown in Equation 3.4 as derived by Kelly [26].

$$\dot{x}_r = [\sin\theta \quad -\cos\theta \quad 0] \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = 0; \tag{3.4}$$

The local frame of the vehicle is assumed to be located in the center of the tracks, which corresponds to the center of the robot chassis, and the instantaneous center of curvature (ICC) when turning in place. No complex maneuvers are used in the simulation; only one velocity can be applied at a time, so this assumption will suffice.

3.2.2 Model

The purpose of the robot kinematic model is to determine the motor velocity setpoints as a function of the desired robot frame velocity.

$$(\dot{\varphi}_R, \dot{\varphi}_L) = f(\dot{x}_r, \dot{y}_r, \dot{\theta}_r, g, r, l) \tag{3.5}$$

where $\dot{\varphi}_R$ and $\dot{\varphi}_L$ are the motor angular velocities, $\dot{x}$ is the lateral velocity, $\dot{y}$ is the forward velocity, $\dot{\theta}$ (or ω) is the angular velocity in the robot frame, g is the gear ratio, r is the track cog radius, and l is the track seperation distance.

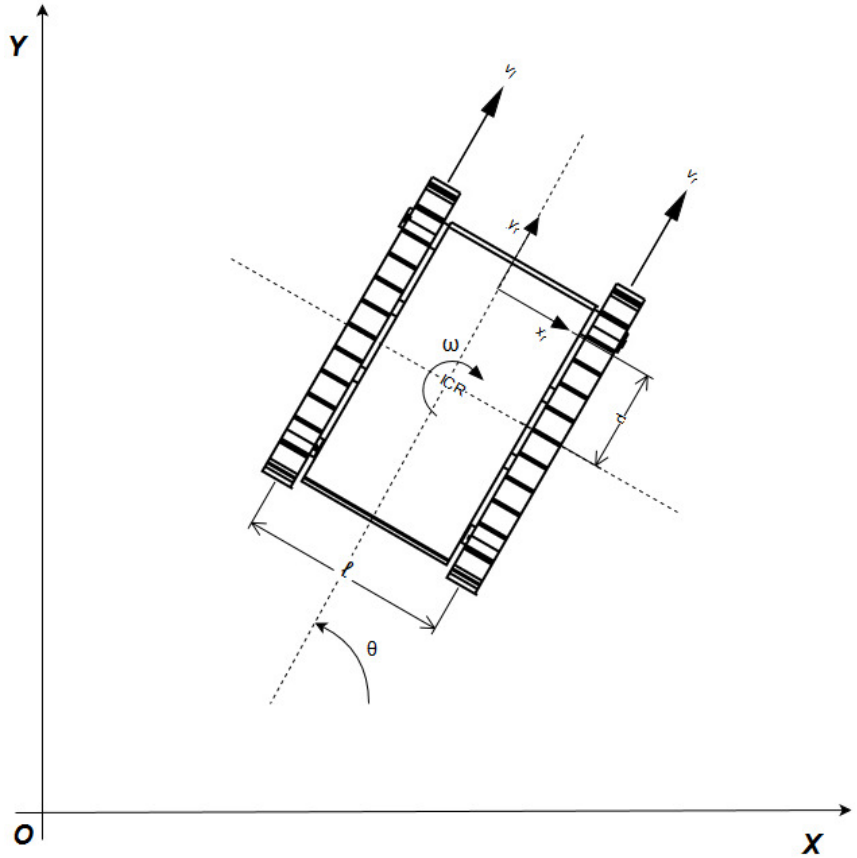


Figure 3.4: *Kinematic Diagram*

Inverse kinematics will be used to convert the local robot frame velocities to the track velocities as shown in Equation 3.6 as derived by Kelly [26].

$$\begin{bmatrix} v_R \\ v_L \end{bmatrix} = \begin{bmatrix} \dot{y}_r + \dot{\theta}_r \frac{l}{2} \\ \dot{y}_r - \dot{\theta}_r \frac{l}{2} \end{bmatrix} = \begin{bmatrix} 1 & \frac{l}{2} \\ 1 & -\frac{l}{2} \end{bmatrix} \begin{bmatrix} \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} \tag{3.6}$$

The motor angular velocities can then be related to the track velocities by rearranging Equation 3.3, thus producing Equation 3.7.

$$\begin{bmatrix} \dot{\varphi}_R \\ \dot{\varphi}_L \end{bmatrix} = \begin{bmatrix} g \frac{v_R}{r} \\ g \frac{v_L}{r} \end{bmatrix} \tag{3.7}$$

The calculated motor angular velocity can then be input into the motor to achieve the desired robot chassis velocity.

3.3 Odometry

Odometry is the estimate of distance travelled by the summation of track displacement at each time step. To achieve odometry, a rotary encoder is mated to the motor, and measures the amount of rotation experienced by the motor. Incremental encoders are typical and output a count value, but in the simulation absolute encoders are employed, providing the angular position. The displacement of each track is calculated as shown in Equation 3.8, where r is the cog radius, φ is the angular position of the motor, and g is the gear ratio. [26]

$$\begin{bmatrix} \Delta s_r \\ \Delta s_l \end{bmatrix} = \begin{bmatrix} \left(\frac{r\varphi_r}{g}\right)_n - \left(\frac{r\varphi_r}{g}\right)_{n-1} \\ \left(\frac{r\varphi_l}{g}\right)_n - \left(\frac{r\varphi_l}{g}\right)_{n-1} \end{bmatrix} \tag{3.8}$$

In order to convert displacement to the change in Cartesian coordinates the displacement and heading change are calculated in Equations 3.9 and 3.10.

$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2} \tag{3.9}$$

$$\Delta \theta = \frac{\Delta s_r - \Delta s_l}{2l} \tag{3.10}$$

With the change in displacement and heading, the change in Cartesian coordinates can be calculated as shown in Equation 3.11. [28]

$$\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \theta \end{bmatrix} = \begin{bmatrix} \Delta s * \cos\left(\theta + \frac{\Delta \theta}{2}\right) \\ \Delta s * \sin\left(\theta + \frac{\Delta \theta}{2}\right) \\ \frac{\Delta s_r - \Delta s_l}{2 * L} \end{bmatrix} \tag{3.11}$$

Since the LIDAR sensor is not mounted over the instantaneous center of rotation of the robot, but offset in the y-axis, it has to be compensated. In order to compensate, the heading of the robot is tracked such that the position can be calculated as if the LIDAR were on an arm rotating about the instantaneous center of rotation (ICR) as shown in Equation 3.12, where d is the distance of the offset in the y-axis.

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} x_{n-1} + ((d \cos \theta)_n - (d \cos \theta)_{n-1}) \\ y_{n-1} + ((d \sin \theta)_n - (d \sin \theta)_{n-1}) \\ \theta \end{bmatrix} \tag{3.12}$$

3.4 Summary

With the chassis and sensors, we now have a platform to maneuver and sense the surroundings. Given the chassis we were able to model the kinematics of the robot and now have the equations to give us corresponding motor velocity values given the desired robot frame velocities.

Chapter 4

Landmark Acquisition

For localization to be plausible, landmarks must be present in the environment, to be used as references for the robot to reference its position when observed. Landmarks are geometric features in the environment that can be easily observed and distinguished such as walls, corners, or static objects in the room. Landmarks must also be capable of being extracted from the LIDAR point cloud repeatedly and accurately.

Landmark acquisition is composed of three major sections. The first describes what constitutes a valid and re-observable landmark. The second describes how the point cloud data points are transformed into meaningful line segments. Lastly, line geometries are evaluated to identify features as potential landmarks from the environment, known as feature extraction.

4.1 Landmark Selection

Landmarks are features in the environment that can be distinguished repeatedly no matter the orientation of the robot. Landmarks are used by the robot to localize itself within its environment, serving as known locations to reference. Choosing the correct landmarks for localization is crucial, and given the beam tunnel floor plan and surroundings, there are few features that can be extracted.

Major factors in considering landmarks include repeatability, accuracy, uniqueness, and frequency; too few landmarks can cause the robot to diverge if error accumulates between landmark observations. Too many can cause misassociations from one landmark to the next. [5]

Beam structures, corners, and the straight walls of the beam tunnel can all serve as static landmarks, can be seen repeatedly, can be accurately extracted, and are frequent enough to prevent the localization strategy from diverging. The beam structures are I-beams that support the beam pipe and associated equipment, and because of their unique shape, differentiating them from the rest of the tunnel is possible. Corners are unique and can be extracted by checking orthogonality and close proximity of the line segments to each other, although they are sparse in the tunnel. The tunnel wall can be used for maintaining the bearing of the robot, which in the extended distance covered in the tunnel, can amount to significant error.

4.2 Line Extraction

To define the observations of features as potential landmarks, we must first convert the LIDAR point cloud into more useful data. Given the landmark selections made, their geometries can be defined by lines, thus the LIDAR point cloud is converted into line segments that can then be evaluated as landmark observations. A typical raw LIDAR scan, with the sensor centered at the origin, can be seen in Figure 4.1.

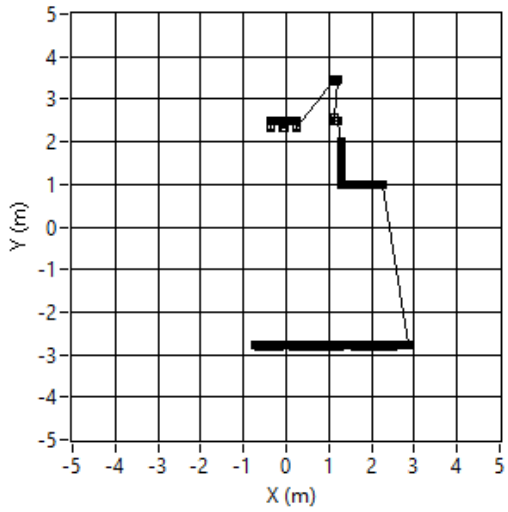


Figure 4.1: *Raw LIDAR Point Cloud*

Prior to processing the data, extraneous LIDAR points are removed to minimize complexity. The Successive Edge Following (SEF) algorithm is then performed to segment the data point cloud into smaller contiguous clusters based on distance differences from point to adjacent point. The Split-and-Merge algorithm is then employed to split segments further, should the perpendicular distance from the first and last points deviate too far. If not, then the algorithm will merge all the points into a single, two-point line segment. These algorithms are described in the following sections 4.2.1 and 4.2.2.

4.2.1 Successive Edge Following

The Successive Edge Following (SEF) algorithm segments the LIDAR point cloud into contiguous clusters of points based on a Euclidean distance threshold between adjacent points [14]. Since the LIDAR output is in polar coordinate form,

the data is converted into SI units and transformed from polar to Cartesian, enabling us to calculate Euclidean distance.

$$x = r * \cos\theta \tag{4.1}$$

$$y = r * \sin\theta \tag{4.2}$$

The improved SEF, as described by Yan [16], incorporates an adaptive threshold distance for segmentation. Yan's method is preferable as the distance between adjacent points becomes greater the further the points extend from the LIDAR. The adaptive threshold distance ΔD is deduced from the mean of the range of the scan data as shown in Equation 4.3 and the angle between points k, which is a constant. The adaptive threshold is formulated as shown in Equation 4.4.

$$\bar{r} = \frac{\sum_{i=1}^{n} r_i}{n}; \tag{4.3}$$

$$\Delta D = k\bar{r} \tag{4.4}$$

Each consecutive point in the data set is iterated through, calculating the Euclidean distance between adjacent points as shown in Equation 4.5. That distance is compared to the adaptive distance threshold ∆D Equation 4.6.

$$D(i, i+1) = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \tag{4.5}$$

$$f(k; j; 0,1) = \begin{cases} D(i, i+1) < \Delta D; \ [x_i, y_i], j+1 \\ D(i, i+1) \geq \Delta D; \ k+1, j = 0 \end{cases} \tag{4.6}$$

If the distance between adjacent points is greater than the adaptive threshold, the cluster is segmented from the point cloud. [14]

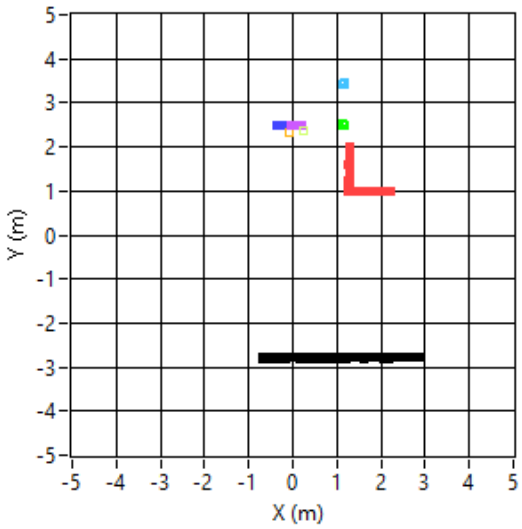


Figure 4.2: *Successive Edge Following Data Point Post Process*

The resulting segmentation from the SEF can be seen in figure 4.2, where each cluster of points can be differentiated by color.

4.2.2 Split-and-Merge

Split-and-merge is a Nlog(N) complex recursive algorithm [12]. Given the output from the SEF is an array of point clusters after segmentation, Split-and-Merge processes each set of points individually. Within each point cluster, a line is fit from the first to last point, and the furthest perpendicular distance is calculated. Perpendicular distance is compared to a distance threshold, if the distance is less than the threshold, the line segment is passed on and the next set is considered. Otherwise the set is segmented at that point and the process is continued recursively until no more splits are performed. After the entire data set

is considered, collinear segments are merged [12, 29]. The Split-and-Merge algorithm can be visualized in Figure 4.3.

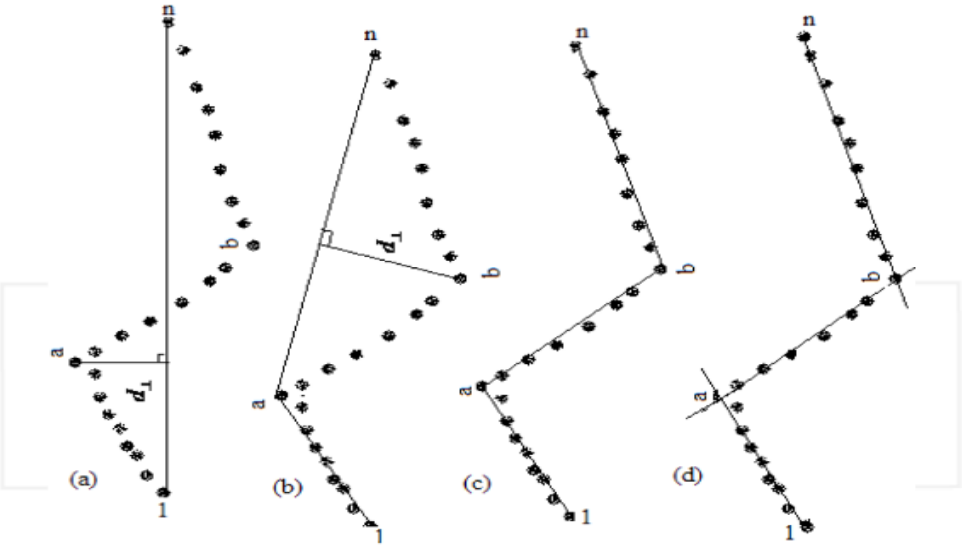


Figure 4.3: *Split-and-Merge Procedure [17]*

Prior to Split-and-Merge, we rotate the point clusters by -π/2 radians such that y-axis positive is bearing origin in the local robot frame using rotation matrix in Equation 4.7.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = R_\theta \begin{bmatrix} x \\ y \end{bmatrix}; where\ R_\theta = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \quad (4.7)$$

Any set that contains a single point is discarded, then the first and last points in the cluster fitted to create a line. The points in the cluster are iterated through, calculating the perpendicular distance; which is the shortest distance between the fitted line and the point in question. [17]

$$perp.dist. = \frac{|(y_2-y_1)x_0-(x_2-x_1)y_0+x_2y_1-y_2x_1|}{\sqrt{(y_2-y_1)^2+(x_2-x_1)^2}} \quad (4.8)$$

Here, the first point is (x_1, y_1), the second point is (x_2, y_2), and the point of interest is (x_0, y_0).

The maximum perpendicular distance is identified along with the point index, to compare with the distance threshold. If the largest distance is greater than the specified tolerance, the line segment is split at this point and the two segmented subarrays are passed back into itself and run recursively until no more splits are made.

From the Split-and-Merge algorithm some remnants exist that are too short for feature identification. To minimize the associated error, Euclidean distance is employed to filter out any line segment that is less than a decimeter in length, simultaneously filtering out any singular points as well.

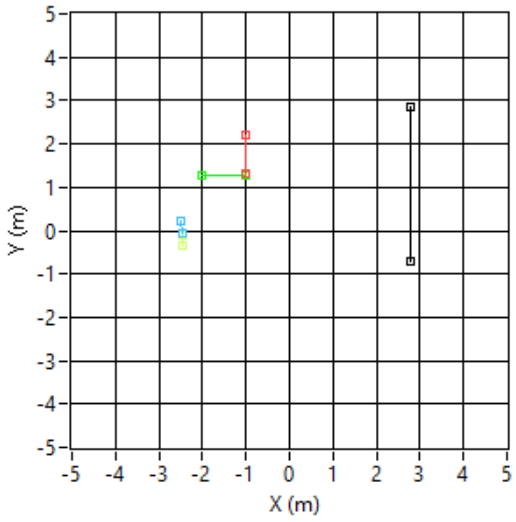


Figure 4.4: *Split-and-Merge Line Segments*

The results from the Split-and-Merge algorithm are shown in figure 4.4, where point clusters from the SEF have been converted into line segments. All three types of features are present in this scan and will be discussed in the following section 4.3 feature extraction.

4.3 Feature Extraction

For a feature to be considered a landmark, it must meet important criteria. First it must be highly visible so that the robot can observe it from any angle. Second, it must be easily determined with simple geometries so that the LIDAR can detect the feature repeatedly and reliably. Lastly, it must be unique such that the feature will not be erroneously detected. In the beam tunnel there are three distinct features that meet these criteria: corners, beam support structures, and the tunnel wall. In this section, we will discuss the methods used to extract these features from the line segments obtained from the previous section.

4.3.1 Corner Extraction

Corners are readily visible by LIDAR scans, and although fairly sparse, present themselves as good landmark candidates. Given that the LIDAR scan has already been processed into line segments, orthogonality can be calculated using dot product as shown in Equation 4.9.

$$\vec{l}_n \cdot \vec{l}_{n+1} = \left\|\vec{l}_n\right\| \left\|\vec{l}_{n+1}\right\| \cos\theta \tag{4.9}$$

Where θ is the angle between $\mathbf{l}_n$ and $\mathbf{l}_{n+1}$, translating to Equation 4.10 using dot product cosine rule, also known as the sliding window corner detection [17].

$$\theta = acos\left(\frac{\vec{l}_n \cdot \vec{l}_{n+1}}{\|\vec{l}_n\|\|\vec{l}_{n+1}\|}\right) \tag{4.10}$$

If the angle between line segments is within 0.03 radians the intersecting point between them is calculated. The intersection point is found using determinants in line-line intersection as shown in Equations 4.11 and 4.12.

$$X_x = \left(\frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}\right) \tag{4.11}$$

$$X_y = \left(\frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}\right) \tag{4.12}$$

The intersection point is the location of the corner feature, as derived from the line segments.

4.3.2 Beam Support Structure Feature Extraction

Beam support structures are located frequently along the tunnel and have a fixed position. The structures are all similar in their construction, I-beams mounted back to back as can be seen in figure 4.5.

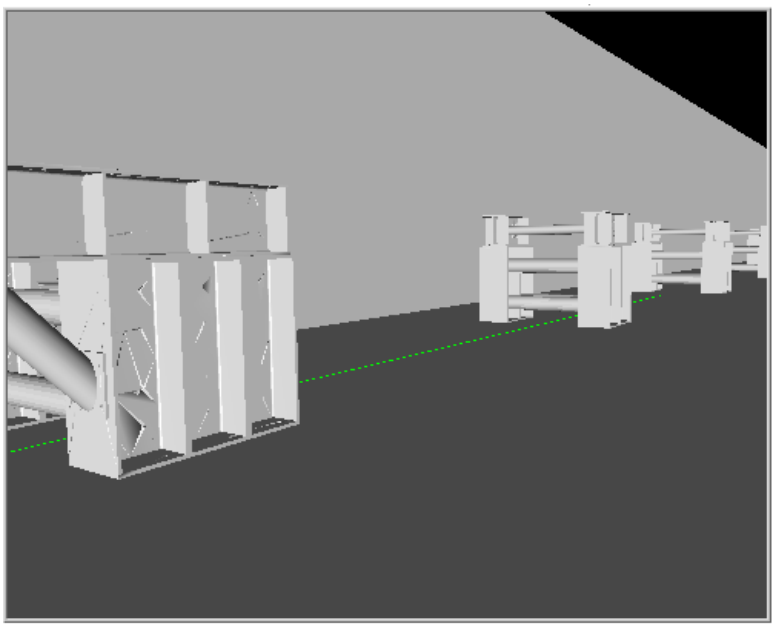


Figure 4.5: *Simulated Beam Support Structure Feature*

The protruding fins of the I-beams do not meet the length requirement set by the Split-and-Merge algorithm and is filtered out. These features therefore resemble consecutive collinear line segments.

To extract the feature, collinearity is checked using the following equality.

$$\overrightarrow{AB} + \overrightarrow{BC} + \overrightarrow{CD} = \overrightarrow{AD} \tag{4.13}$$

Where $\overrightarrow{AB}$ is the first line segment, and $\overrightarrow{CD}$ is the second line segment representing the beam structure as can be seen in figure 4.6.

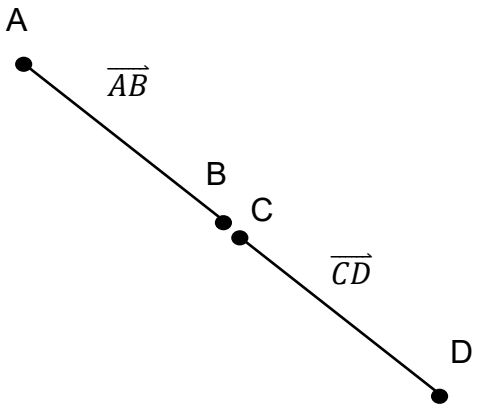


Figure 4.6: *Beam Support Structure Line Segment Geometry*

If the sum is approximately equal to $\overrightarrow{AD}$, a point is interpolated at the midway point between the two-line segments, representing the beam support structure position and is accepted as the feature.

4.3.3 Wall Feature Extraction

Because of the long distances travelled by the robot, heading error causes significant Abbe error to accumulate, which contributes to the overall positional error. Abbe error is the magnified positional deviation over a travelled distance due to heading error, and can be calculated in Equation 4.14. [30]

$$\varepsilon = d\ \tan\theta \tag{4.14}$$

Where ε is the abbe error, d is the distance travelled, and θ is the heading error of the robot as depicted in figure 4.6.

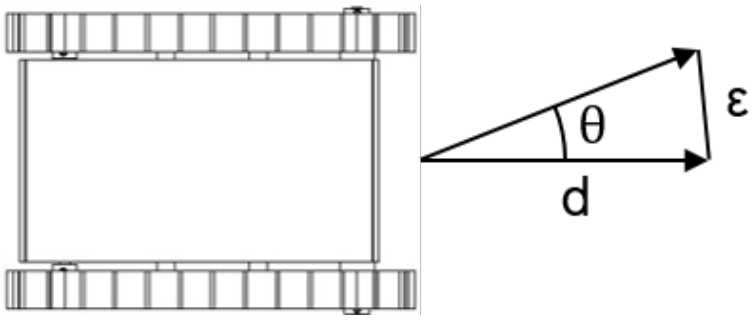


Figure 4.7: *Abbe Error Illustration*

Because the tunnel contains a long straight wall on one side of the beam tunnel, it is used to constrain the Abbe error accumulation, providing an accurate reference for the bearing of the robot. To accomplish this, the line segments need to represent the wall with high confidence and spurious line segments need to be removed from consideration. A series of validation gates are established to ensure the spurious segments are not selected, which will cause divergence. The first gate ensures the line segment is greater than three meters from robot origin in the world map, which in the global map there are no other geometries other than the tunnel wall from that distance from origin. The second gate checks the length of the line segment by calculating the Euclidean distance from Equation 4.5 from point to point, and ensures it exceeds 2 meters before acceptance. The line segment length is important because in the tunnel there is not a geometry with a line segment length greater than 2 meters outside of the tunnel wall. Between the two criteria, spurious segments are avoided and the line segment is associated to the wall correctly.

If the line segment is believed to represent the wall, the robot bearing θ is calculated using the atan2 function as shown in Equation 4.15.

$$\theta = atan2(y_2 - y_1, x_2 - x_1) \tag{4.15}$$

The bearing is held with high confidence and replaces the EKF prediction for bearing as it was found to not be as accurate or frequent enough to prevent the accumulated Abbe error.

4.4 Summary

From the LIDAR point cloud we were able to produce high accuracy line segments that represent the beam tunnel using the SEF and Split-and-Merge algorithms. It was found that these algorithms allow for some erroneous line segments that had to be filtered out to promote robust feature extraction by limiting line segment length.

By running the LIDAR in the simulation, features were selected by what could be identified easily and repeatedly. From that knowledge, geometric principles were applied to confirm the features as represented by line segments.

Chapter 5

Localization

Following the successful implementation of landmark acquisition and kinematics, the Extended Kalman Filter (EKF) state estimation technique can be exploited to achieve localization when paired with a data association algorithm. Data association is an important step for maintaining accurate robot position. To achieve data association, the Individual Compatibility Nearest Neighbor (ICNN) algorithm is implemented as it works well with the EKF, utilizing similar principles. The process is broken up into three primary steps: the prediction step of the EKF, ICNN data association, and the measurement update of the EKF. The processes of the EKF and ICNN algorithms will be discussed in this chapter.

5.1 Extended Kalman Filter

The Extended Kalman Filter (EKF) is a state estimation algorithm for non-linear systems and consists of two primary steps: a prediction based on the motion model, and a measurement update based on the measurement model. [6, 8, 9] Figure 5.1 shows the block diagram representing the EKF.

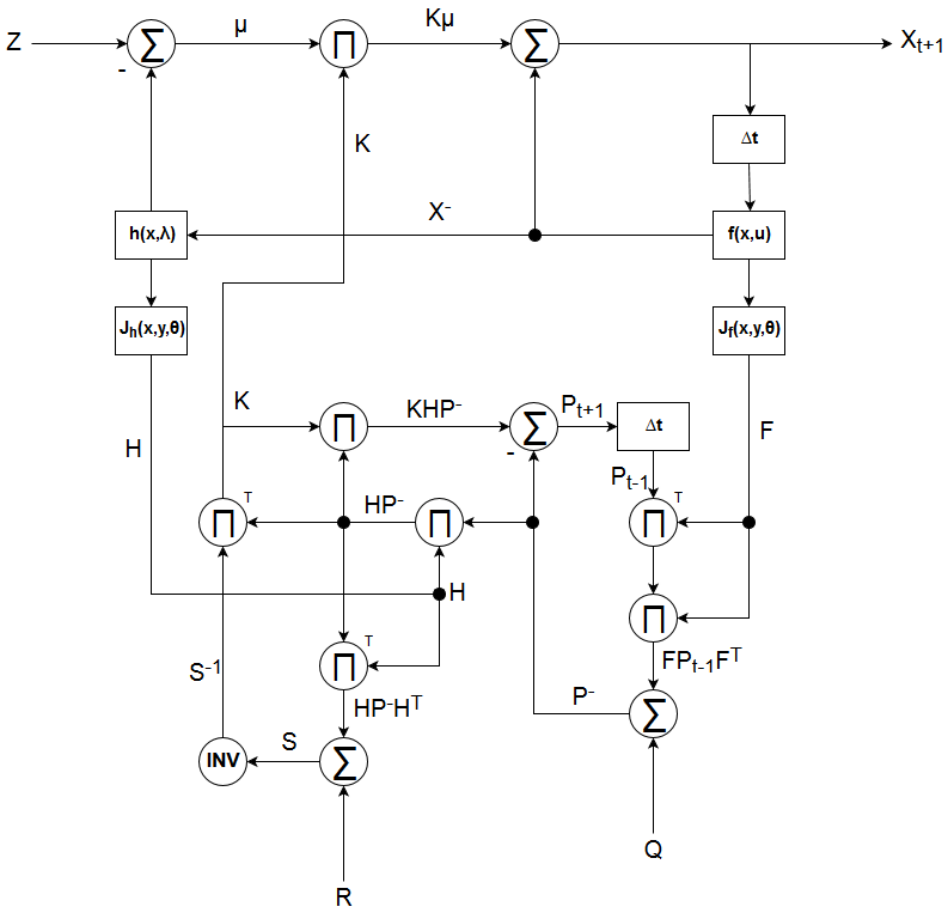


Figure 5.1: *Extended Kalman Filter Block Diagram*

The EKF is a filter in the sense that it filters, or minimizes the Gaussian noise and error in estimating the current state of the robot, linearizing the state transition and measurement model about the mean and covariance of the current state. The EKF became a prime candidate for localizing in the beam tunnel as it relies on a map for the measurement model as a reference. Given that the beam tunnel is a structured environment, landmark locations are easily identified as presented in chapter 4.

5.1.1 Initialization

For the EKF to execute effectively, the initial conditions must be well known. The initial conditions consist of the state matrix, and the error covariance matrix. The state matrix consists of the robot position, heading, and landmark positions. The state matrix is a 3+2n size column matrix where the first three rows represent the robot state, and consecutive pairs are landmark positions and can be seen in Equation 5.1. [5]

$$X = \begin{bmatrix} x_r \\ y_r \\ \theta_r \\ x_i \\ y_i \end{bmatrix} \tag{5.1}$$

The estimation error covariance matrix must be initialized as well, and is set to be a 3+2n x 3+2n size identity matrix where n is the number of landmarks. The matrix is set to identity so that the robot position and landmark locations are independent initially and correlated as the robot observes landmarks.

5.1.2 Motion Update

The motion model from chapter 3, which produces the change in the robot state using odometry, is used for the robot's prediction step. From there we can derive the predicted state transition matrix $f(X, u)$.

$$f(X,u) = X^{-} = \begin{bmatrix} x + \Delta x \\ y + \Delta y \\ \theta + \Delta\theta \end{bmatrix} = \begin{bmatrix} x + \Delta s\, cos\theta \\ y + \Delta s\, sin\theta \\ \theta + \Delta\theta \end{bmatrix} \tag{5.2}$$

To incorporate the predicted state transition matrix, it needs to be linearized. To linearize, the Jacobian of the state transition matrix $f(X,u)$ needs to be calculated. The state transition Jacobian derivation is shown in Equation 5.3. [8]

$$F = J_f(x,y,\theta) = \begin{bmatrix} \frac{\delta X_1}{\delta x} & \frac{\delta X_1}{\delta y} & \frac{\delta X_1}{\delta \theta} \\ \frac{\delta X_2}{\delta x} & \frac{\delta X_2}{\delta y} & \frac{\delta X_2}{\delta \theta} \\ \frac{\delta X_3}{\delta x} & \frac{\delta X_3}{\delta y} & \frac{\delta X_3}{\delta \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -sin\theta\Delta s \\ 0 & 1 & cos\theta\Delta s \\ 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

The process noise Q, where C is a representation the odometry accuracy, is formulated using Equation 5.4. [5]

$$Q = WCW^T = \begin{bmatrix} C\ \Delta x^2 & C\ \Delta x\Delta y & C\ \Delta x\Delta\theta \\ C\ \Delta y\Delta x & C\ \Delta y^2 & C\ \Delta y\Delta\theta \\ C\ \Delta\theta\Delta y & C\ \Delta\theta\Delta y & C\ \Delta\theta^2 \end{bmatrix} \tag{5.4}$$

$$where\ W = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix}$$

Next, the estimation error covariance matrix P becomes a crucial element to the EKF procedure as it contains the covariance of the robot position, landmark positions, the covariance between the robot and landmarks, and the covariance between landmarks. The estimation error covariance matrix is built systematically. The upper 3x3 matrix is the robot position covariance, represented by P^{rr}, and is updated by the odometry model as shown in Equation 5.5.

$$P^{rr} = FP^{rr}F + Q \tag{5.5}$$

The robot-landmark cross covariance P^{ri} is located in the upper 3 rows of the matrix, and is found using Equation 5.6. [9]

$$P^{ri} = FP^{ri} \tag{5.6}$$

The covariance matrix P is symmetric and is defined in Equation 5.7. [9]

$$P^{-} = \begin{bmatrix} P^{rr} & P^{ri} \\ P^{ir} & P^{ii} \end{bmatrix} \tag{5.7}$$

Where P^{ii} is the covariance of the landmark positions. Details on the state transition can be found in references. [5, 8, 9]

5.1.3 Data Association

To reliably associate the robot's sensor data to its environment, a couple of data association methods were fused. First that of individual compatibility nearest neighbor (ICNN), which is a widely used technique in robotics, and an optimization algorithm developed in order to reconsider the best choice given the ICNN criteria.

5.1.3.1 Individual Compatibility Nearest Neighbor

The derivation of the ICNN algorithm is based on comparing the extracted features to the landmark map using Mahalonobis distance and then passing it through a Chi-Squared validation gate which deems them compatible. The ICNN

is an n x m complex algorithm, where n is the number of landmarks in the map, and m is the number of visible features. [21]

To initiate the ICNN algorithm, the measurement model must first be established. The measurement model h for a range bearing model, where (λ_x, λ_y) are the landmark locations, and is defined in Equation 5.8. [31]

$$h = \begin{bmatrix} range \\ bearing \end{bmatrix} = \begin{bmatrix} \sqrt{(\lambda_x - x)^2 + (\lambda_y - y)^2} \\ \tan^{-1}\left(\frac{\lambda_y - y}{\lambda_x - x}\right) - \theta \end{bmatrix} \tag{5.8}$$

Taking the Jacobian of the predicted measurement model gives us H, which tells us how the range and heading change with respect to the robot state (x, y, θ).

$$H = \begin{bmatrix} \frac{\delta r}{\delta x} & \frac{\delta r}{\delta y} & \frac{\delta r}{\delta \theta} \\ \frac{\delta b}{\delta x} & \frac{\delta b}{\delta y} & \frac{\delta b}{\delta \theta} \end{bmatrix} = \begin{bmatrix} \frac{x-\lambda_x}{r} & \frac{y-\lambda_y}{r} & 0 \\ \frac{\lambda_y - y}{r^2} & \frac{\lambda_x - x}{r^2} & -1 \end{bmatrix} \tag{5.9}$$

The innovation μ is defined by the difference between the predicted landmark locations h from Equation 5.10 and the LIDAR measurement z.

$$\mu = z - h \tag{5.10}$$

The measurement noise R reflects the range and bearing error associated with the LIDAR sensor. The measurement noise is represented by a 2x2 diagonal matrix with the upper left value representing the range multiplied by the percent error c in the distance, and the lower diagonal value representing the bearing error. [5]

$$R = \begin{bmatrix} rc & 0 \\ 0 & bd \end{bmatrix} \tag{5.11}$$

Before the Mahalonobis distance between the measured features and landmark map can be calculated, the innovation covariance must first be computed using the Jacobian of the measurement model H, error covariance P, and the measurement noise R.

$$S = HP^{-}H^{T} + R \tag{5.12}$$

The squared Mahalonobis distance can now be defined using the innovation and associated covariance as shown in Equation 5.13. [21]

$$D_{ij}^{2} = \mu^{T}S^{-1}\mu \tag{5.13}$$

Being that the Mahalonobis distance follows a Chi-Squared distribution, we compare the calculated distance to a Chi-Squared distributed variate with two degrees of freedom with a 95% cumulative probability α. [32]

$$D_{ij}^{2} < \chi_{d,\alpha}^{2} \tag{5.14}$$

If the validation gate is satisfied, the landmark-feature pair is passed to be a highly probable match. The validation gate can pass more than one feature to one landmark if features are dense or spurious. [33]

5.1.3.2 Compatibility Optimization

Because more than one feature can be associated to one landmark using the ICNN, additional consideration is necessary. To rectify any spurious associations, an optimal selection algorithm was developed. Optimization can

have a similar complexity to that of the ICNN, but typically is much smaller, as no more than a couple of features are typically individually compatible with a landmark.

Since the ICNN is a greedy algorithm, it never reconsiders potential matches, instead accepting the first acceptable match which can be problematic. In the developed algorithm, multiple feature matches are reconsidered such that the minimum Maholanobis distance is taken as the associated feature to the landmark in question.

5.1.4 Measurement Update

The robot state estimate obtained from the motion model is not reliable due to the error attributed to the odometry. In order to compensate for these errors, the landmark measurements associated to the landmark map is used to correct the state.

The Kalman gain K is calculated as a weighted compensation between the motion model prediction and measurement update.

$$K = P^{-}H^{T}(S)^{-1} \tag{5.15}$$

The robot state update can now be computed using the Kalman gain and innovation. The process is repeated for each feature that has been matched to a landmark using the ICNN algorithm, updating the feature map based on the observations and motion model.

$$X_{t+1} = X^{-} + K\mu \tag{5.16}$$

Lastly, the error covariance is updated to reflect the measurement update.

$$P_{t+1} = (I - KH)P^{-} \tag{5.17}$$

The EKF state estimation localization has been defined, and is now repeated for each subsequent LIDAR scan, constantly updating the robot state. Details on the measurement update can be found in references [34, 35, 36].

5.2 Summary

A state estimation algorithm has been applied in the form of the EKF coupled with the ICNN data association algorithm. The EKF creates error from the odometry data due to slipping of the differential drive, and from poor bearing estimation from the measurement update of the EKF due to low landmark density in the tunnel. Tuning of the EKF to more heavily rely on the LIDAR measurements was instituted, but was found to not be accurate enough to reduce the bearing error to an acceptable level where Abbe error would not accumulate. By neglecting the EKF bearing update and using the wall as a bearing landmark when observed, error was reduced to a level where the EKF would no longer diverge.

Chapter 6

Simulation

In this chapter the LabVIEW robotics module simulator and function library are examined. The vehicle model, feature acquisition, localization concepts and techniques are implemented using this software. Results from the simulator testing is presented and the effectiveness of the techniques selected is reviewed.

6.1 LabVIEW Robotics

Due to availability and timing constraints in the beam tunnel, it was decided that simulating the localization algorithm would be the best path forward to provide validation and feasibility of deployment during beam production. Given that LabVIEW software was already being used in conjunction with a National Instruments cRIO to control the robot and the ability to go from the simulation to hardware with few changes to the software, made the LabVIEW robotics module simulator an attractive option. Therefore, it was selected as the robot simulation package for the project. Included in the package is a robotics simulation environment and function library.

The LabVIEW robotics simulator is based on the Open Dynamics Engine [37], a physics-based library for simulating rigid body dynamics. The simulator consists of three main aspects that allows the testbed to resemble the beam tunnel

environment; a CAD model importer, robot model builder, and the robotics environment simulator wizard.

Prior to using the simulator, CAD models had to be created of both the beam tunnel and the robot, which was done using SolidWorks. To import the CAD models of the beam tunnel and the robot, unnecessary detail had to be removed and the polygonal mesh had to be decimated to keep the file size limited and allow the simulation to run efficiently. The CAD model importer was then used to insert the robot and beam tunnel environment models into the LabVIEW robotics simulator. The robot model builder allowed the addition of the robot chassis selected, including the LIDAR sensor, joint definition for the differential drive components, and an origin of axis. The environment in which the robot will operate is also incorporated using the robotics environment simulator wizard, where axes are specified and obstacles (similar to what would be seen in the tunnel during normal operation) are added.

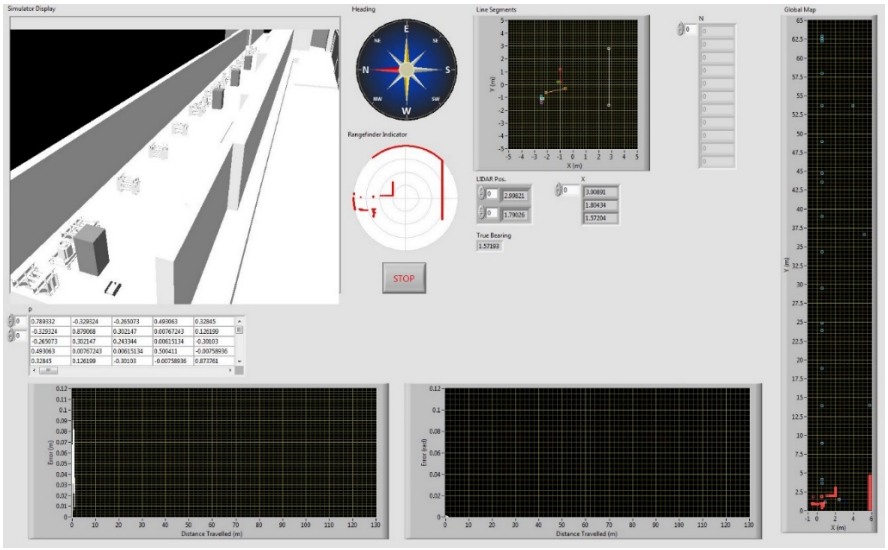


Figure 6.1: *LabVIEW Robotics Simulation Front Panel*

In addition to the simulator, the robotics module includes robotics specific function libraries, facilitating rapid development of robot design [38]. Because of the nature of the proposed localization strategy, all of the algorithms used had to be developed from the ground up.

6.2 Simulation Results

Results of the LabVIEW robotics simulation will be discussed in this section, and errors in the results will be addressed and analyzed. The EKF tested was incorporated as discussed in chapter 5 with the sensors discussed in chapter 3.

To obtain results, the robot was operated through an environment similar to the beam tunnel as it would during a typical deployment. A cross section of the beam tunnel can be seen

As the robot traversed the beam tunnel it was able to consistently extract features, and correctly associate them to the landmark map, thus updating the robot state with relatively little error.

In order to gauge the effectiveness of the localization algorithm, Euclidean distance is calculated between the estimated robot position and the ground truth along with the absolute difference in heading as described by Chen and shown in Equations 6.1 & 6.2. [8]

$$Error_p = \sqrt{\left(x_{r,e} - x_{gt}\right)^2 + \left(y_{r,e} - y_{gt}\right)^2} \quad (6.1)$$

$$Error_\theta = \left|\theta_{r,e} - \theta_{gt}\right| \quad (6.2)$$

As can be seen in Figure 6.2 the positional error varies due to the odometry error and increases in areas where landmarks are sparse. At the end of the sector, where landmarks are dense, the positional error decreased significantly. For the entire run, the greatest positional error accumulated was less than 0.3 meters.

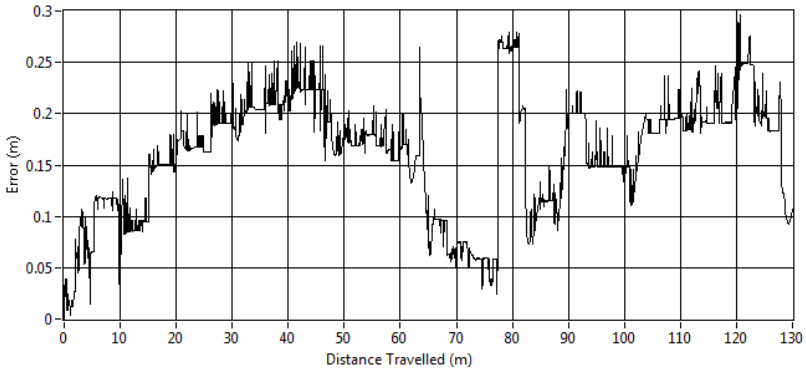


Figure 6.2: *Positional Error*

The heading error is minimized by using the walls in the tunnel as described in section 4.3.3. Without the wall extraction algorithm added to the EKF the heading would diverge, contributing to the Abbe error in the robot position, causing the EKF to diverge further. As shown in figure 6.3 the heading error grows when the robot maneuvers 180 degrees at the end of the tunnel and no longer observes the wall with the LIDAR as a reference but is then corrected once the wall is observed again.

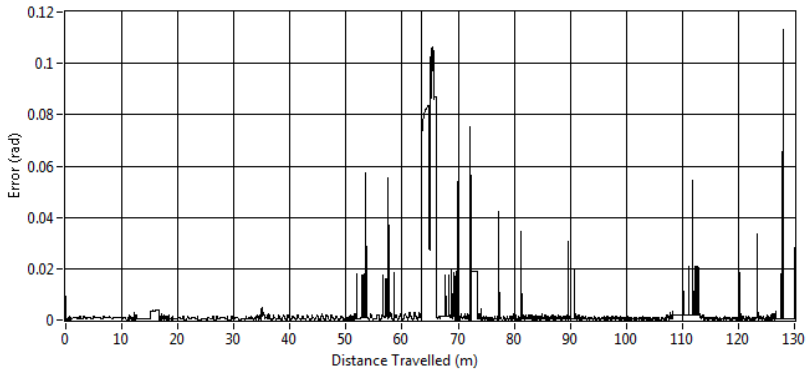


Figure 6.3: *Heading Error*

In the simulation, true robot position is known, which allows for algorithm performance measurements as the estimated position can be measured against the actual position providing absolute error for the robot state. The robot path (blue) and the robot's ground truth (green) is graphed in figure 6.4 using Equations 6.1 and 6.2 to show the accuracy of the EKF as a localization method in a mapped representation. Along with the robot path, the landmarks are displayed to show how error grows when the robot is relying purely on odometry data.

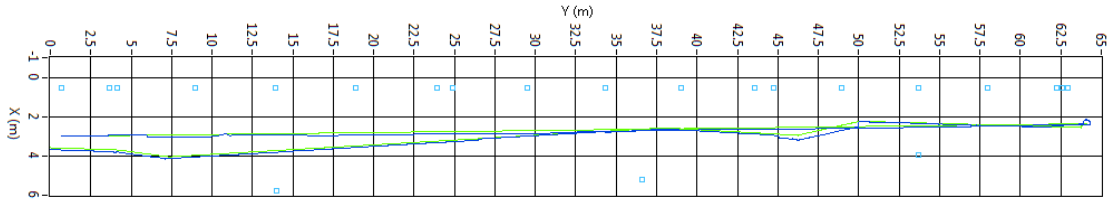


Figure 6.4: *Ground Truth vs. Estimated Tracked Position*

Chapter 7

Conclusion

The LabVIEW robotics simulator is used to demonstrate the viability of self-localization in a known environment using LIDAR. The strategy and implementation of the EKF, landmark extraction, and data association is reviewed and the path forward for future work is discussed.

7.1 Summary

The objective of this thesis was to develop a self-localization strategy for a differential drive robot in the LANSCE LINAC beam tunnel. Given the knowledge about the environment in the tunnel, a decision had to be made to determine the direction of the method of which localization was to be realized. After extensive research and following various paths, the extended Kalman filter was chosen as the best choice for the application as the positional estimator.

Implementation of the EKF as a state estimator for the robot in the simulator shows potential with ideal conditions and well-known initial conditions. The robot would show error accumulation between landmarks as they are fairly sparse in the tunnel. It was also found that the EKF is sensitive to the heading of the robot in maintaining its position regardless of landmark density. The Abbe error is credited to the poor odometry accuracy of a differential drive robot during turning

maneuvers due to slippage. Heading error was kept to a minimum by making the wall an absolute reference regardless of what the EKF estimated. With the Abbe error minimized, the robot was capable of tracking its position with relatively low error without diverging.

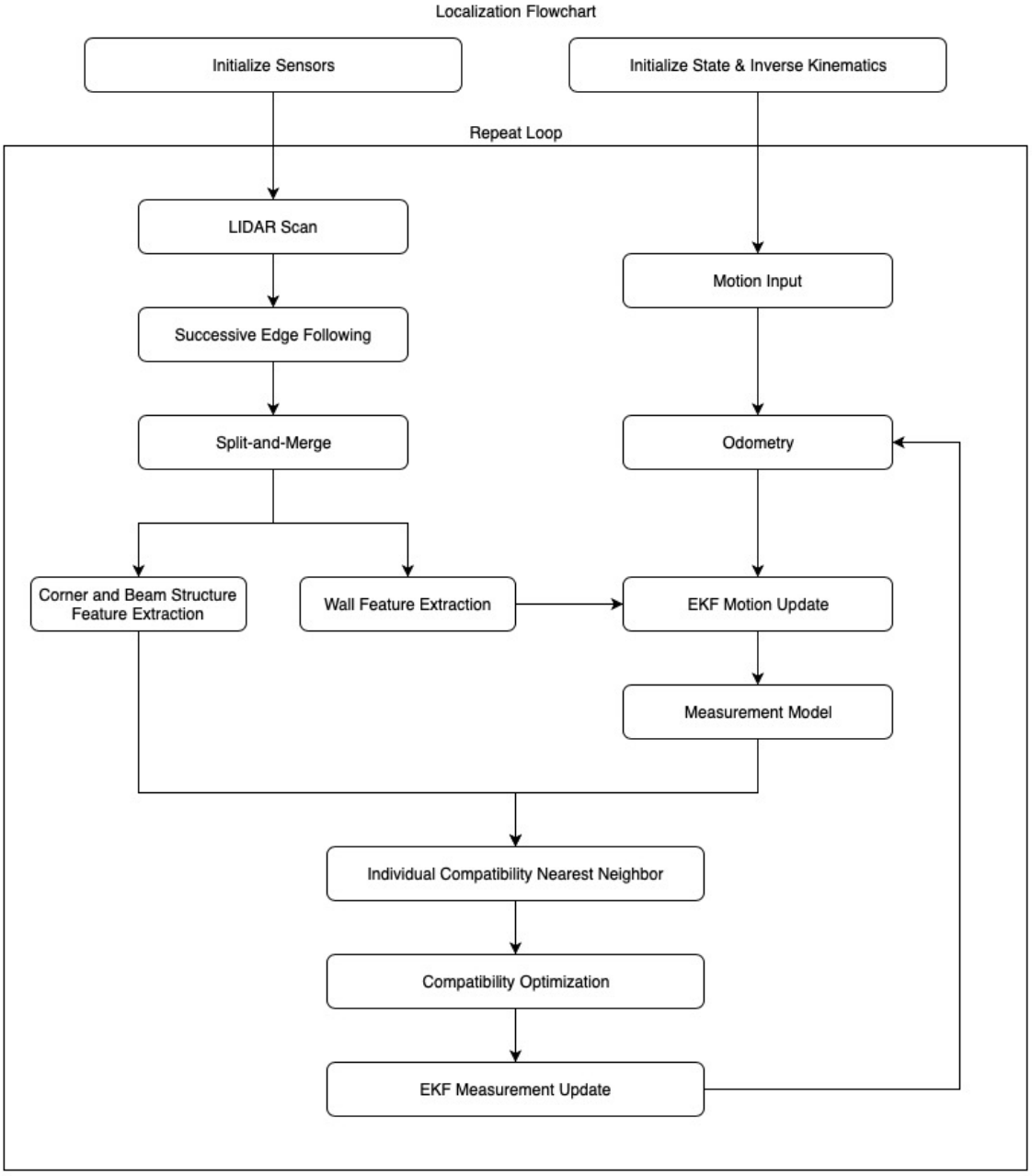
7.2 Future Work

With promising simulation results using the proposed strategy for localization, testing in the tunnel on hardware would be the desired next step. Because the simulated environment was stripped down, and relatively obstacle free, the actual tunnel will need to be mapped to provide an accurate reference for the EKF to use when implemented.

Future work will also include studies for facilitating full robot autonomy. Navigation and obstacle avoidance studies are the two main focuses in achieving autonomy past localization. In parallel with realizing autonomy, additional beam diagnostic sensors will need to be studied in order to maximize the potential to provide substantial measurements in determining beam status to operators as well.

Appendix A

Localization Flowchart



Bibliography

[1] K.F. Schoenberg and P.W. Lisowski, “LANSCE A Key Facility for National Science and Defense,” *Los Alamos Science*, no. 30, 2006.

[2] H. Watkins, D. Baros, D. Martinez, L Rybarcyk, J. Sedillo, and R. Valicenti, “Upgrades to the LANSCE Isotope Production Facilities Beam Diagnostics,” *Proceedings of IBIC2016,* Barcelona, Spain, 2016.

[3] C. Keatmanee, J. Baber, and M. Bakhtyar, “Simple Example of Applying Extended Kalman Filter,” *First International Electrical Engineering Congress*, Thailand, March 2014.

[4] P. Jensfelt, *Approaches to Mobile Robot Localization in Indoor Environments.* PhD thesis, Royal Institute of Technology, Stockholm, Sweden, 2001.

[5] S. Riisgaard and M.R. Blas, “Slam for Dummies,” *A Tutorial Approach to Simultaneous Localization and Mapping*, vol. 22, June, pp. 1-127, 2004.

[6] J. Sola, “Simultaneous Localization and Mapping with the Extended Kalman Filter,” unpublished. http://www.joansola.eu/JoanSola/eng/JoanSola.html

[7] H. Hu and D. Gu, “Landmark Based Navigation of Industrial Mobile Robots,” *International Journal of Industry Robot*, vol. 27, no. 6, pp. 458-467, 2000.

[8] L. Chen, H. Hu, and K. McDonald-Maier, “EKF Based Mobile Robot Localization,” *Emerging Security Technologies (EST) Third Internationl Conference*, Sept. 2012.

[9] N.A. Othman and H. Ahmad, “The Analysis of Covariance Matrix for Kalman Filter Based SLAM with Intermittent Measurement,” *Proceedings of the 2013 International Conference on Systems, Control and Informatics*, 2013.

[10] H.R. Everett, *Sensors for Mobile Robotics Theory and Application*. Wellesley Massachusetts: A K Peters, Ltd., 1995.

[11] M. Lacroix, J. Santos, and R Stiffler, “Advantages of Magnetic Encoder Technology in Harsh Operating Environments,” TIMKEN, North Canton, Ohio, *Technical Report*, Aug. 2011. https://www.timken.com/pdf/10416_EncodersWhitePaper.pdf

[12] V. Ngyen, A. Martinelli, N. Tomatis, and R. Siegwart, “A Comparison of Line Extraction Algorithms using 2D Laser Rangefinder for Indoor Mobile Robotics,” *In Proceedings of Conference on IROS 2005*, Edmonton, Canada, 2005.

[13] A. Siadat, A. Kaske, S. Klausmann, M. Dufaut, and R. Husson, “An Optimized Segmentation Method for a 2D Laser Scanner Applied to Mobile Robot Navigation,” *In Proceedings of the 3rd IFAC Symposium on Intelligent Components and Instruments for Control Applications,* 2007.

[14] Y. Bu, H. Zhang, H. Wang, R. Liu, and K. Wang, "Two Dimensional Laser Feature Extraction Based on Improved Successive Edge Following," *Applied Optics*, May 2015.

[15] J. Lv, Y. Kobayashi, A.A. Ravankar, and T. Emaru, "Straight Line Segments Extraction and EKF-SLAM in Indoor Environment," *Journal of Automation and Control Engineering,* vol. 2 no. 3*,* Sept. 2014.

[16] M. Namoshe, O. Matsebe, and N. Tlale, "Feature Extraction: Techniques for Landmark Based Navigation System," *Intech*, 2010.

[17] G.A. Borges, and M.J. Aldon, "A Split and Merge Segmentation Algorithm for Line Extraction in 2-D Range Images," *In Proceedings of 15th International Conference on Pattern Recognition*, Barcelona, Spain, Sept. 2000.

[18] A. Cooper, *A Comparison of Data Association Techniques for Simultaneous Localization and Mapping,* MS thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2005.

[19] P.C. Mahalanobis, "On the Generalized Distance in Statistics," *In Proceedings of the National Institute of Sciences of India,* vol. 2, 1936.

[20] R. Gnanadesikan and J.R. Kettenring, "Robust Estimates, Residuals, and Outlier Detection with Multiresponse Data," *Biometrics,* Vol. 28, No. 1, pp. 81-124, Mar. 1972.

[21] J.A. Castellanos, J. Neira, and J.D. Tardos, "Map Building and SLAM Algorithms," *Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications*, Lewis, New York, New York, 2006.

[22] Dr. Robot, "Jaguar V2 User Guide," *Dr. Robot Inc*, V.28.08.18, Available: http://jaguar.drrobot.com/images/Jaguar_V2_Manual.pdf

[23] A. Bentley, B. Marohn, J. Mason, and B. Wooton, *Beam Tunnel Monitoring Robot Final Design,* thesis, New Mexico Tech, Socorro, NM, 2012.

[24] Kamitani, Maeda, Mori, and Yamamoto, "Scanning Laser Range Finder URG-04LX-UG01 Specifications," *Hokuyo Automatic Co. LTD*, June 2009.

[25] R. Siegwart and I.R. Nourbakhsh*, Introduction to Autonomous Mobile Robotics,* Cambridge, Massachusetts: MIT Press, 2004.

[26] A. Kelly, *Mobile Robotics Mathematics, Models, and Methods*, Cambridge University Press, 2014.

[27] R. Dhaouadi and A.A. Hateb, "Dynamic Modelling of Differential Drive Mobile Robots using Lagrange and Newton-Euler Methodologies: A Unified Framework," *Advances in Robotics and Automation,* vol. 2, 2013.

[28] J.L. Crowley and P. Reignier, "Asynchronous Control of Rotation and Translation for a Robot Vehicle," *Journal of Robotics and Autonomous Systems,* Feb. 1993.

[29] S. Yuan, L. Huang, F. Zhang, Y. Sun, and K. Huang, “A Line Extraction Algorithm for Mobile Robot using Sonar Sensor,” *Proceeding of the 11th World Congress on Intelligent Control and Automation,* Shenyang, China, June, 2014.

[30] R. Leach, “Abbe Error/Offset,” *CIRP Encyclopedia of Production Engineering,* 2014.

[31] T.Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, “Consistency of the EKF-SLAM Algorithm,” *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2006.

[32] Y. Bar-Shalom and T.E. Fortmann, *Tracking and Data Association*, Academic Press In., 1988.

[33] J. Neira and J.D. Tardos, “Data Association in Stochastic Mapping using the Joint Compatibility Test,” *IEEE Transactions on Robotics and Automation*, vol. 17, Issue 6, pp 890 – 897, Dec. 2001.

[34] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, Cambridge, Massachusetts: MIT Press, 2006.

[35] H. Durrant-Whyte and T. Bailey, “Simultaneous Localization and Mapping,” *IEEE Robotics & Automation,* June 2006.

[36] M.R. Nepali, D.A.H. Prasad, S. Balasubramaniam, V. EN, and Ashutosh, “A Simple Integrative Solution for Simultaneous Localization and Mapping,” *International Journal of Robotics and Automation,* vol. 5, Issue 2, 2014.

[37] “Overview of the LabVIEW Robotics Simulator,” *National instruments,* 22 December 2015, http://www.ni.com/white-paper/14133/en/

[38] “Overview of the LabVIEW Robotics Module,” *National Instruments,* 24 August 2016, http://www.ni.com/white-paper/11564/en/